Amendments to the Claims

1. (Currently amended) A computer system for managing memory for an executing object-oriented program, the program comprising a plurality of objects and the system comprising:

a processing unit;

a computer-readable memory partitioned into a plurality of regions, each region containing at least one object in the memory; and

at least one data structure in the computer-readable memory representing at least one region ~~shape graph~~ shape graph describing relations between the regions, the relations based at least in part on references between objects contained in the regions;

wherein the processing unit is configured such that given a reference to a target object, the processing unit can identify the partitioned region of memory containing the target object ~~can be identified~~ by using information from the at least one region shape graph.

2. (Original) The system of claim 1, wherein:

the at least one region shape graph comprises a plurality of nodes connected by edges;

each node represents a region in memory; and

each edge represents one or more references between the objects of one region and the objects of another region.

3. (Original) The system of claim 2, wherein using information from the shape graph comprises:

identifying a first region in memory which contains an object which has a reference to the target object;

identifying a first node in the at least one shape graph which represents the identified first region;

identifying the edge leading from the identified first node which represents the reference to the target object;

identifying a second node which the identified edge leads to; and

identifying the region which is represented by the second node as the region containing the target object.

4. (Original) The system of claim 1, wherein each region has a shape-graph associated with it and each shape-graph is stored along with the regions it is associated with.

5. (Original) The system of claim 1, further comprising a memory-management software module configured to:

determine, for a region, when no objects contained in the region are referenced by any fields outside of the region; and

delete the region upon making such a determination.

6. (Original) The system of claim 5, wherein configured to determine when no objects are referenced comprises configured to:

keep a count for each region of the number of references made to objects contained in the region; and

upon determining that the count for a region is zero, determine that no objects contained in the region are referenced by any other fields.

7. (Original) The system of claim 5, further comprising a garbage collector.

8. (Original) The system of claim 1, where at least one shape graph represents fewer than the total number of regions.

9. (Original) The system of claim 1, where no more than one region parameter is passed to a method for every object passed to a method.

10. (Currently Amended) A method for compiling an object-oriented program, the program configured to be executed in a system employing region-based memory-management and the method comprising:

receiving source code for an object-oriented program;

performing a points-to analysis on the source code to develop at least one data structure containing region association metadata for the program;

adding instrumentation to the program, the instrumentation configured:

to cause objects to be created in regions based on information in the data structure; and

to cause deletion of all objects in a region when a determination is made that no objects in the region are referenced by any fields outside the region; ~~and~~

compiling the program to produce executable code for the program; and

storing the executable code in a computer-readable storage medium.

11. (Original) The method of claim 10, wherein the data structure containing region association metadata comprises a shape graph.

12. (Original) The method of claim 11, wherein performing a points-to analysis comprises:

creating alias sets for parameters based on statements contained in the methods of the object-oriented program;

unifying the alias sets based on statements contained in the methods of the program;

creating at least one shape graph with nodes defined by alias sets and edges defined by field mappings between alias sets; and

associating the nodes of the shape graph with potential memory regions.

13. (Original) The method of claim 11, wherein the added instrumentation at least in part comprises:

region-creation code which creates a region given a shape-graph;

object-allocation code which, given a region and object information, allocates an object within certain region; and

region-lookup code which, given a region and an identifier of a field, identifies the region referenced by that field.

14. (Original) The method of claim 13, wherein the added instrumentation further comprises field-setting code which, given two regions and a reference to a field from an object in one region which references an object in the other region, sets the shape graph to connect the nodes corresponding to the two regions through an edge corresponding to the field.

15. (Original) The method of claim 13, wherein employing region-based memory management comprises at least in part:

keeping a count for each region or set of regions of the number of references made to the objects contained in that region or set; and

upon determining that the count for a region or set of regions is zero, deleting the region or set; and

wherein the added instrumentation further comprises:

incrementing code which, given a region or set of regions, increases the count kept for that region or set; and

decrementing code which, given a region or set of regions, decreases the count kept for that region or set.

16. (Currently Amended) A computer-readable storage medium containing instructions, which, when executed, cause a computer to compile an object-oriented program to be executed in a system employing region-based memory management by performing the following process:

receiving source code for an object-oriented program;

performing a points-to analysis on the source code to develop at least one shape-graph template for the program;

adding instrumentation to the program, the instrumentation configured:

to cause objects to be created in regions based on the shape-graph template; and

to cause deletion of all objects in a region when a determination is made that no objects in the region are referenced by any objects outside the region; and

compiling the program.

17. (Original) The computer-readable medium of claim 16, wherein performing a points-to analysis comprises:

creating alias sets for parameters based on statements contained in the methods of the object-oriented program;

unifying the alias sets based on statements contained in the methods of the program;

creating at least one shape graph with nodes defined by alias sets and edges defined by field mapping between alias sets; and

associating the nodes of the shape graph with potential memory regions.

18. (Original) The computer-readable medium of claim 16, wherein the added instrumentation at least in part comprises:

region-creation code which creates a region given a shape-graph;

object-allocation code which given a region and object information, allocates an object within certain region; and

region-lookup code which, given a region and an identifier of a field, identifies the region referenced by that field.

19. (Original) The method of claim 18, wherein the added instrumentation further comprises field-setting code which given two regions and a reference to a field from an object in one region which references an object in the other region, sets the shape graph to connect the nodes corresponding to the two regions through an edge corresponding to the field.

20. (Original) The method of claim 18, wherein employing region-based memory management comprises at least in part:

keeping a count for each region or set of regions of the number of references made to the objects contained in that region or set; and

upon determining that the count for a region or set of regions is zero, deleting the region or set; and

wherein the added instrumentation further comprises:

incrementing code which, given a region or set of regions, increases the count kept for that region or set; and

decrementing code which, given a region or set of regions, decreases the count kept for that region or set.